REMARKS

These remarks are responsive to the Final Office Action, dated March 6, 2008. Currently, claims 1-20 are pending with claims 1, 8 and 13 being independent. Claims 1-3, 5, 7-8, and 13 have been amended to expedite prosecution of this application to allowance and to overcome Examiner's objections. Claims 16-20 have been added. Support for these amendments is found in the Applicants' specification at least on page 8, lines 3-13; page 9, line 21 to page 10, line 27; page 14, line 24 to page 18, line 19.

35 U.S.C. 112

In the Final Office Action, the examiner rejected claim 1 under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. The Examiner stated that "wherein based on the list of files and the list of repository nodes stored at the destination fileserver, a replica of a file in the list of files is recovered form a repository node in the list of repository nodes" is not described in the specification. Applicants respectfully disagree and traverse this rejection.

Applicants refer the Examiner to the following sections within the specification that disclose the above-referenced element:

> "By having data replicated to a local and at least one remote repository from the originating fileserver, these repositories act as a replacement for traditional on-site and off-site tape storage systems and tape vaulting services. According to one embodiment, in the event of a site disaster, all fileservers that were lost are recovered by deploying new fileservers at a surviving site and recreating the content of the failed fileservers from the content in the surviving repositories." (Specification, Page 6, line 24 to Page 7, line 2).
>
> "By maintaining this metadata in replicated repositories, fileservers and/or their shares may be easily migrated to another fileserver. In one embodiment metadata for a file includes the following information:
> - The fileserver name where the file was created
> - The size of the file
> - A list of all of the repository nodes that maintain a replica of this file
> - A computed MD5 content checksum of the file when it was first created or last modified." (Specification, Page 8, lines 4-13)

"• As a background task, the Recovery service 45 shown in FIG9 also repopulates fileserver Z with all of the files that were present in their full form on fileserver Y before the share migration was requested.
• Finally, the share or shares that were migrated from fileserver Y to fileserver Z are deleted from fileserver Y since they are no longer referenced by client applications from that server." (Specification, Page 10, lines 20-26).

"• Protection Management – Number of replicas per repository. This feature allows a storage administrator to determine how many replicas 80 of data to create within each repository 81 when a share is protected. In one embodiment, there must be at least one replica stored in a repository that is local to the share's fileserver. It's possible to maintain multiple replicas within a single repository. In this case, replicas are maintained on different repository nodes of a repository to ensure continued access to a replica in the event of a single repository node failure or network failure. The location and number of replicas can be changed over time. To increase data availability for data that is increasing in criticality, more replicas per repository and additional repositories may be specified. For data that is decreasing in importance, fewer replicas may be maintained in the repositories, which makes more storage capacity available to other shares that are also using those repositories." (Specification, Page 12, lines 10-25).

"According to embodiments of the invention, all input/output activity initiated by client file requests is intercepted by the filter driver 31. The fileserver software maintains a list of all modified or created files since this last snapshot occurred. In one embodiment, snapshot intervals can range from 15 minutes to 24 hours, based on the backup frequency 19 of the protection policy. On the schedule of the backup frequency, the mirror service 33 prepares all modified files in a share to be put into the repositories 8 (shown in Fig. 2) that are specified in that share's protection policy. The protection policies are stored and replicated across multiple repositories, and they are cached and regularly updated within each fileserver in the protection policy cache 34. For example, if a share's protection policy has its backup frequency set to one hour, on the transition to the next hour, the mirror service 33 initiates a backup of all changed files in the last hour to a local repository 28. For all new files, any repository node of the local repository can be used to hold a replica of a file. For files that have been modified, the mirror service directs new versions of the existing file to the same repository node as prior versions of that file. The mirror service queries the location cache 36 to determine which repository node should receive an updated version of an existing file. This location cache is updated regularly by the location manager 35 when the fileserver writes files to specific repository nodes. Once the location manager identifies all destination repository nodes for each file of a share for the latest collection of updated or created files, the fileserver communicates to each local repository via a fileserver API 37 and a repository node API 38." (Specification, Page 14, line 23 to Page 15, line 18).

"FIG10 illustrates one embodiment of modules that implement a process that protects data to one or more remote repositories to completely protect client data from site disaster. FIG10 displays a local repository node 46 that, from the actions described in FIG9, holds the first replica of data. FIG10 also shows a remote repository node 47. These are connected to each other across a metropolitan or wide-area network 48. In one embodiment, all data that is transferred between local and remote repositories may be secured by virtual private networking (VPN) 49 encryption. The local repository node's replication service 50 is responsible for reviewing the protection policy 51 for all files that were just created as part of the recent fileserver backup. Each repository node acts as a peer of other repository nodes. Based on the protection policy each repository node manages the movement of files among all repository nodes using repository node APIs 52, 53, data movers 54, and file transfer modules 55, 56. Once the data is replicated to remote repositories, the location manager 57 of each repository node updates the location cache 58 to track where files are maintained within that repository node. The version service 59 of the remote repository node manages file version compression, and compaction according to the protection policy." (Specification, Page 16, lines 6-25).

"During the load of the relevant stub files, the recovery/migration service updates 64 the fileserver's location cache to maintain a list of the repository nodes that are associated with each recovered file. Once these stub files are in place, clients can be connected to the new fileserver to allow them to immediately view all of their files. To the clients, the recovery operation appears to be complete, but accesses to each file will be slightly slower since a file requested by a client initiates a transfer from a repository node back into the fileserver.

FIG11 also shows the second phase of the two phases of migration/recovery. In the second phase, clients have access 65 to their files. The recovery service responds to client requests for files as a high priority task and replaces the stub file. e.g., a 4KB stub file, with the full content of each file requested. At a lower priority, the recovery service begins/continues loading the files that were most-recently accessed on the previous fileserver back into the destination fileserver. This process expedites client requests for data while the migration/recovery process proceeds." (Specification, Page 17, lines 13-30).

"FIG13 shows one embodiment of the recovery service 45 of FIG9. With reference to FIG13, a receiving component 94 receives metadata and stub files associated with the set of files at a destination fileserver. A location updating component 95, in communication with the receiving component, maintains a list of repository nodes that are associated with each file in the set of files. In addition, a stub file replacement component 96, in communication with the receiving component, replaces the stub files with the full content of the file associated with the stub file as described above." (Specification, page 18, lines 12-20).

> "A protection policy defines how client data within the fileserver's primary disk storage system will be protected among the collection of repositories located in multiple data centers. Each share's protection policy specifies, among other things, the number of replicas to maintain within multiple repositories. In one embodiment, replicas reside in at least two repositories. These repositories should be located a disaster-safe distance from each other." (Specification, Page 21, lines 23-29).

The above-referenced specification paragraphs are provided here for exemplary purposes only and intended to demonstrate adequate disclosure support for claimed element of "wherein based on the list of files and the list of repository nodes stored at the destination fileserver, a replica of a file in the list of files is recovered form a repository node in the list of repository nodes" recited in claim 1. As such, the rejection of claim 1 is respectfully traversed. The Examiner is requested to reconsider and withdraw his rejection of claim 1.

35 U.S.C. 101

In the Final Office Action, the Examiner stated that he interprets the term "fileserver" as "excluding transmission media, signals, or any form or energy, such that the claim clearly falls within a statutory class of invention as required under the terms of 35 U.S.C. 101." (Final Office Action, page 3).

It does not appear that the Examiner was rejecting any particular claim or objecting to a specific term within the claims. As such no traversal is required. Applicants respectfully point out to the Examiner that the term "fileserver" is discussed in the specification as follows:

> "A fileserver is typically configured to have tens of shares. These shares allow the primary storage capacity of the fileserver to be shared and securely partitioned among multiple client systems." (Specification, Page 7, lines 6-9).

The Examiner is respectfully referred to Applicants' Specification for any clarification of terms recited in the claims of the present application.

35 U.S.C. 103(a)

In the Final Office Action, the Examiner rejected claims 1-4, 8-12, and 16 under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent Publication No. 2004/0049513 to Yakir

(hereinafter, "Yakir") in view of U.S. Patent Pub. No. 2002/0055972 to Weinman JR (hereinafter, "Weinman"). In the Final Office Action, the Examiner stated that Yakir discloses all elements of claim 1 except that it does not "explicitly disclose 'maintaining, at the destination fileserver, a list of repository nodes that contain a replica of each file in the set of files, and a list of files in the set of files stored at the destination fileserver;' 'initiating recovery of files in the set of file on the destination fileserver, wherein based on the list of file and the list of repository nodes stored at the destination fileserver, a replica of a file in the list of files is recovered from a repository node in the list of repository nodes'." (Final Office Action, page 5). The Examiner stated that Weinman discloses this element. The Examiner further stated that it "would have been obvious to a person of ordinary skill in the art at the time the invention was made to have applied Weinman's disclosure above to the disclosure of Yakir for the purpose of backing up the file in case of a disaster occurs. Doing so, would provide recovery for the migrated file in Yakir, in case the server results in accidental deletion, disaster that destroys data, and loss of location of which Weinman resolves (see 0051)." (Final Office Action, page 6). Applicants respectfully disagree and traverse this rejection.

Amended claim 1 recites, *inter alia*, receiving, at a destination fileserver, a set of stub files associated with the set of files; maintaining, at the destination fileserver, a list of repository nodes that contain a replica of each file in the set of files, and a list of files in the set of files stored at the destination fileserver; initiating recovery of files in the set of files on the destination fileserver, wherein based on the list of files and the list of repository nodes stored at the destination fileserver, a replica of a file in the list of files is recovered from a repository node in the list of repository nodes. The initiating further includes using a stub file in the set of stub files, allowing access to a full content of a file associated with the stub file; and, replacing each stub file with a full content of the file associated with the stub file. The replacing includes receiving a client request for a specified file in the set of files; replacing the stub file associated with the specified file with a full content of the specified file; and, replacing remaining stub files in the set of stub files with respective full contents of remaining files in the set of files while replacing the stub file associated with the specified file with the full content of the specified file.

As understood by Applicants, Yakir relates to techniques for moving a stub file from an **originating storage location** to a **destination storage location** without recalling the migrated

data corresponding to the stub files. (emphasis supplied). (Yakir, Abstract). Yakir moves stub files from an originating storage location to a destination storage location without recalling migrated data corresponding to the stub file. (Yakir, para. [0009]). Yakir's originating and destination storage locations can be on the same storage unit and assigned to the same or different file servers. (Yakir, para. [0009]). It appears that Yakir discloses an advanced Hierarchical Storage Management ("HSM") based storage system that allows shares of data to be migrated from an originating server to a destination server. As Applicants pointed out in their previous responses and during June 25, 2008 interview, one of the major drawbacks of Yakir is that it requires that both originating and destination servers are present in order to migrate files. Yakir **is incapable of performing any data migration** when originating server is **non-operational, failed and/or non-existent**. (emphasis supplied). This is in contrast to the present invention that provides disaster recovery operation when originating fileserver is non-operational, failed and/or non-existent. In the Final Office Action, the Examiner asserts that there "is no need in Yakir for two fileservers. See Figure 3" and that "claim 1 directed to transferring and recovery of files." (Final Office Action, page 29). Yakir's FIG. 3 is reproduced below for Examiner's reference:

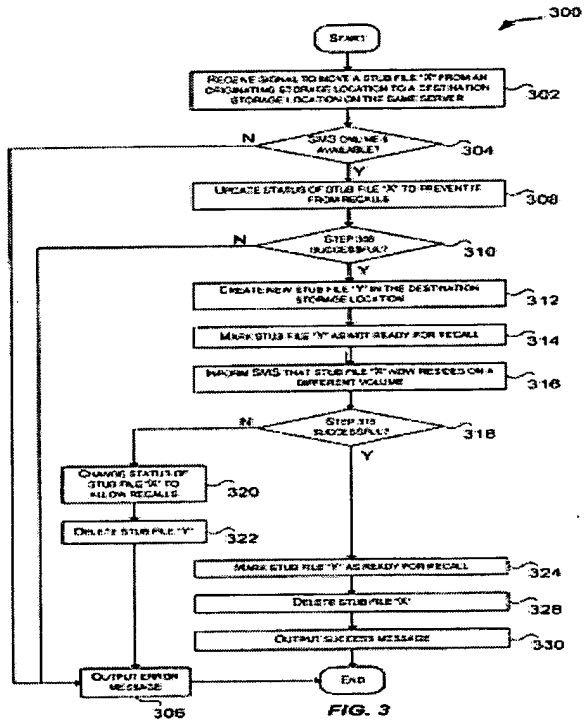

FIG. 3

Yakir's FIG. 3 refers to "originating storage location" and "destination storage location" located on the "same server". (Yakir, FIG. 3, "302"). As such, failure of Yakir's "same server" would render FIG. 3 inoperative and thus, inapplicable to the present invention's claims. In contrast, the present invention will always maintain a destination fileserver, even if disaster occurs with the originating fileserver, since the destination fileserver is where a set of stub files associated with the set of files is received, as recited in claim 1. Yakir's FIG. 3, showing the "same server" for originating and destination locations, does not address this situation. Yakir further discloses an "originating server" to which an originating volume is allocated and a "destination server" to which stub files from the originating volume are moved. (Yakir, paras. [0040]-[0045]). Thus, Yakir requires presence of both servers to move its storage volumes. If Yakir's originating server fails, nothing will be moved. This is different from the present invention, which, as stated above, maintains a destination fileserver. One of the advantages of the present invention is that in the event originating fileserver has failed due to a disaster, for example, its content can always be recovered at a destination fileserver. Such operation is not possible using Yakir. As such, Examiner's assertion is improper.

In the Final Office Action, the Examiner asserted that "claim [1] only requires that remaining s[t]ub files be replaced. There is nothing regarding non-requested files. It would be obvious for the user to request remaining stub files to be replaced by merely running the method again." (Final Office Action, page 30). Applicants respectfully disagree. Yakir simply replaces a requested stub file (using file location information stored in the stub files) with its full content, but does not replace other or remaining files while replacing the stub file requested by the client, which is contrary to the recitation of claim 1. Yakir's disclosure and Examiner's assertion purport to describe an extremely cumbersome method of file replacement, i.e., one file at a time and only upon request by the user. The present invention provides a highly efficient method of replacing a file based on a user request and replacing remaining files on the destination fileserver **while** user requested file is being replaced. (emphasis supplied). Clearly, Yakir fails to disclose this element of claim 1. As such, in addition to the Examiner's admission that Yakir fails to disclose "maintaining" and "initiating" steps (Final Office Action, page 5), Yakir also fails to disclose, teach or suggest the above referenced elements of claim 1.

Weinman fails to cure the deficiencies of Yakir. As understood by Applicants, Weinman discloses a data dispersion method for reducing a risk of losing a file by replicating it across geographically-distant locations. (Weinman, paras. [0014]-[0016]). Upon creation of an object, Weinman mirrors the object to n-1 additional mirror sites in the network. The number of copies may change upon creation of additional copies of disasters occurring in the additional sites. (Weinman, para. [0015]). Weinman also determines whether too few or too many copies have been created and either adds or deletes copies of files. (Weinman, para. [0015]). Weinman's creation and deletion of copies is based on a particular location of such copies, provided that such locations satisfy a particular "distance separation criteria". (Weinman, para. [0015]). In the Final Office Action, the Examiner refers to Weinman's FIGS. 2-5 and asserts that these figures along with Weinman's para. [0040] provide disclosure of claim 1's "maintaining" step. (Final Office Action, page 33). Weinman's FIGS. 2-5 are reproduced below for Examiner's reference.

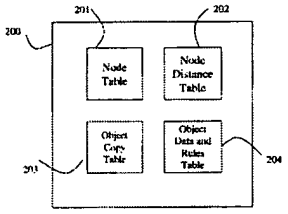

Figure 2

Node Table

Node	Capacity	Location
New York	2 Terabytes	124 5th Ave.
Los Angeles	1 Terabyte	43 6th Ave.
Chicago	1.5 Terabyte	348 7th Ave.

Figure 3

Node Distance Table (miles)

	NewYork	Los Angeles	Chicago
New York	--	2462	719
Los Angeles	2462	--	1749
Chicago	719	1749	--

Figure 4

Object Copy Table

Object	Copy #	Location
Presentation.ppt	1	New York
Presentation.ppt	2	Chicago
Presentation.ppt	3	Los Angeles
Spreadsheet.xls	1	Chicago

Figure 5

Clearly, none of these figures purport to show that a **destination fileserver**, where a set of stub files is received, maintains a list of repository nodes containing a replica of each file **and** a list of files stored at the destination fileserver, as recited in claim 1. (emphasis supplied). Instead, Weinman's FIG. 2 shows a central index server that contains four tables having information about:

1) capacity of the node and location of the node;

2) distance between each node;

3) each data object and location information for each data object;

4) each data object's number of copies, maximum and minimum of copies, size, etc. (Weinman, paras. [0040]-[0046]). None of Weinman's Tables and Weinman's discussion in paras. [0040]-[0046] refer to maintenance of the specific lists recited in claim 1. As stated above, Weinman is concerned with specific geographical separation between copies and a particular

number of copies. Thus, Examiner's assertion is improper. Weinman's FIGS. 3-5 and para. [0015] also do not disclose the "initiating" step recited in claim 1 for at least the reasons stated above with regard to the "maintaining" step, contrary to the Examiner's assertion in the Final Office Action. (Final Office Action, page 33). As previously stated, Weinman is concerned with distance-separating of copies of its files, rather than recovery operations recited in claim 1. Thus, Weinman fails to disclose all elements of claim 1 and does not render it obvious.

There is no motivation to combine Yakir and Weinman. One having ordinary skill in the art having the knowledge of Yakir and its stub file moving techniques would not look to Weinman to resolve its deficiency of requiring the presence of an originating server, as well as, lack of maintaining, at the destination fileserver, a list of repository nodes that contain a replica of each file in the set of files, and a list of files in the set of files stored at the destination fileserver, initiating recovery of files in the set of files on the destination fileserver, wherein based on the list of files and the list of repository nodes stored at the destination fileserver, a replica of a file in the list of files is recovered from a repository node in the list of repository nodes, where the initiating further includes using a stub file in the set of stub files, allowing access to a full content of a file associated with the stub file, and, replacing each stub file with a full content of the file associated with the stub file, where the replacing includes receiving a client request for a specified file in the set of files, replacing the stub file associated with the specified file with a full content of the specified file, and replacing remaining stub files in the set of stub files with respective full contents of remaining files in the set of files while replacing the stub file associated with the specified file with the full content of the specified file, as recited in claim 1. Thus, the combination of Yakir and Weinman fails to yield predictable results and does not render claim 1 obvious.

Claims 2-4, 8-12 and 16 are patentable over Yakir, Weinman, and their combination for at least the reasons stated above with regard to claim 1. As such, the rejection of claims 2-4, 8-12 and 16 is respectfully traversed. The Examiner is requested to reconsider and withdraw his rejection of claims 2-4, 8-12 and 16.

In the Final Office Action, the Examiner rejected claims 5-7, 13-15, and 17-20 under 35 U.S.C. 103(a), as being unpatentable in view of various combinations of Yakir, Weinman, U.S. Patent Pub. No. 2002/0111929 to Pudipeddi et al. (hereinafter, "Pudipeddi"), U.S. Patent No.

5,991,753 to Wilde (hereinafter, "Wilde"), U.S. Patent Pub. No. 2002/0069280 to Bolik et al. (hereinafter, "Bolik"). Applicants respectfully traverse these rejections.

Claims 5-7, 13-15, and 17-20 are patentable over the combination of Yakir and Weinman for at least the reasons stated above with regard to claim 1.

Pudipeddi fails to cure the deficiencies of Yakir and Weinman. As understood by Applicants, Pudipeddi relates to a technique for recalling data objects stored on a media. (Pudipeddi, Abstract). Pudipeddi further discloses a scheduling algorithm that creates "active" and "non-active queues" of tape (or removable disks) read requests, where "active" queue allows retrieval of migrated files and "non-active" queue does not. (Pudipeddi, para. [0006]).

As previously stated in Applicants' prior responses Wilde fails to cure the deficiencies of Yakir and/or Weinman. As understood by Applicants, Wilde discloses a file management system for implementing special handling of files for migration, compression, encryption and logging access to files. (Wilde, Abstract). Wilde distinguishes between a resident file, which is a file with all of its contents stored on a local disk, and a non-resident file, which is a file that has been migrated. (Wilde, Col. 7, lines 35-39). During migration, a list of all the files in the file system that are eligible for migration along with migration attributes is generated. The files are ranked by weighing their age and size factors and then sorted according to the ranking. The resulting list is called a candidate list. (Wilde, Col. 8, lines 11-21).

Bolik also fails to cure the deficiencies of the Yakir and Weinman references. As understood by Applicants, Bolik relates to a mechanism of managing an HSM system that includes an HSM server and a file server having a managed file system. Bolik scans the file system only until a certain amount of migration candidates have been found. Further, Bolik waits until one of two events to happen, namely until a specified wait interval expires or until an auto migration process starts. The candidate determination process resumes the file system scan at the point where it stopped a previous scan and continue to look for migration candidates, again until a certain amount of candidates has been found.

However, the various combinations of Yakir, Weinman, Pudipeddi, Wilde, and/or Bolik fail to disclose, teach or suggest maintaining, at the destination fileserver, a list of repository nodes that contain a replica of each file in the set of files, and a list of files in the set of files stored at the destination fileserver, initiating recovery of files in the set of files on the destination

fileserver, wherein based on the list of files and the list of repository nodes stored at the destination fileserver, a replica of a file in the list of files is recovered from a repository node in the list of repository nodes, where the initiating further includes using a stub file in the set of stub files, allowing access to a full content of a file associated with the stub file, and, replacing each stub file with a full content of the file associated with the stub file, where the replacing includes receiving a client request for a specified file in the set of files, replacing the stub file associated with the specified file with a full content of the specified file, and replacing remaining stub files in the set of stub files with respective full contents of remaining files in the set of files while replacing the stub file associated with the specified file with the full content of the specified file, as recited in claim 1.

Applicants incorporate herein by reference in their entireties and reiterate their arguments submitted in their October 31, 2007 and July 7, 2008 Responses. Applicants further point out some of the advantages of the present invention over the cited references. The present invention is directed to an accelerated fileserver disaster recovery process that can be used in any one of three instances:

1. When a fileserver fails, and the recovery server is at the same site as the failed fileserver

2. When a fileserver is lost during a site disaster, and a remote fileserver is used as a recovery fileserver

3. When one or more shares on a fileserver need to be migrated to another fileserver in order to load-balance file access / performance across multiple file servers

The present invention includes fileservers that incorporate HSM technology in order to retain only the most active files on the more expensive disk storage within the file server tier of storage. The cited references fail to address the above advantages.

Further, in the present invention, a repository is made up of servers that have lower cost disk storage technology. These repository servers maintain all of the backup history for all of the files created or modified on the file server(s). The latest historical version of the backup data in the repository for each fileserver file can also be leveraged to represent the staged-out copy of a file that has been deemed inactive by the fileservers' HSM policy. Inactive files on the file server

can be "stubbed out" to reduce storage capacity consumption on the more costly file server disk subsystem, and these stub files can be pointed to the latest version of that file's backup data.

Because present invention integrates file servers with integrated backup and HSM, a unique operational feature is enabled that is not possible with simple file servers, or HSM-only or backup-only systems, such as the systems disclosed in the cited references. When a fileserver has completely failed (unrecoverable hardware failure or site disaster), the contents of that entire fileserver can be recreated from the backup data in the local or remote repository. In addition, because the file server incorporates HSM, the recovery of data to the recovery file server can be accelerated using a two stage recovery operation.

- Stage 1 repopulates the recovery file server with just HSM stub files. This reduces file server rebuild times from more than a day to less than an hour.
- Stage 2 of the process allows these stub files that are requested by users to be staged in from the repository at high priority. In the absence of requests for files from users, all of the files that were cached in the fileserver that failed are re-cached on the recovery file server as a background task.

Clearly, the cited references fail to provide one skilled in the art with the advantages that are apparent in the present invention.

Hence, claims 2-20 are patentable over the various combinations of Yakir, Weinman, Pudipeddi, Wilde, and/or Bolik for at least the reasons stated above with regard to claim 1. As such, the rejections of claims 2-20 are respectfully traversed. The Examiner is requested to reconsider and withdraw the rejections of claims 2-20.

CONCLUSION

No new matter has been added. The claims currently presented are proper and definite. Allowance is accordingly in order and respectfully requested. However, should the Examiner deem that further clarification of the record is in order, we invite a telephone call to the Applicants' undersigned attorney to expedite further processing of the application to allowance.

Applicants believe that no additional fees are due with the filing of this Amendment. However, if any additional fees are required or if any funds are due, the USPTO is authorized to charge or credit Deposit Account Number: 50-0311, Customer Number: 35437, Reference Number: 25452-015.

Respectfully submitted,

Date: March 27, 2009

Boris A. Matvenko, Reg. No. 48,165
MINTZ LEVIN COHN FERRIS
GLOVSKY & POPEO, P.C.
Chrysler Center
666 Third Avenue, 24th Floor
New York, NY 10017
Tel: (212) 935-3000
Fax: (212) 983-3115